WHAT IS CLAIMED IS:

1. A method comprising:

providing an interface to access a plurality of peripheral devices independent of specific features of the peripheral devices, the interface having a plurality of generic routines commonly shared by the peripheral devices;

upon receipt of a request, calling the generic routines as a function of specific features of a requested one of the peripheral devices; and

causing a native driver of the requested one of the peripheral devices to execute.

2. The method of claim 1, wherein the providing an interface includes:

providing a plurality of parameters to define the specific features of the peripheral devices; and

providing a plurality of native drivers to control the peripheral devices.

3. The method of claim 2, wherein the calling the generic routines includes:

determining from the request the specific features of the requested peripheral device;

calling the generic routines with the parameters of the requested peripheral device; and

using the called routines to access the native driver corresponding to the requested peripheral device.

4. The method of claim 1, further comprising:

upon the execution of the driver, accessing the requested peripheral device.

5. The method of claim 1, further comprising:

upon receipt of another request, using the interface to call the generic routines as a function of specific features of another requested one of the peripheral devices.

6. The method of claim 1, wherein the peripheral devices include a printer, a scanner, an imager, a smart card reader, and a barcode reader.

7. The method of claim 1, wherein the request is a request from an application to connect to the requested peripheral device.

8. The method of claim 1, wherein the request is a request from an application to disconnect from the requested peripheral device.

9. The method of claim 1, wherein the request is a request from the requested peripheral device to connect to an application to provide the application with data acquired by the requested peripheral device.

10. The method of claim 1, further comprising:

providing an emulator to simulate access to the peripheral devices in order to test the interface.

11. The method of claim 1, further comprising:

providing a graphical user display to allow a user to select the peripheral devices to be accessible by the interface; and

providing native drivers corresponding to the selected peripheral devices.

12. A method comprising:

providing a connection class to include generic routines to connect to peripheral devices independent of device-specific features of the peripheral devices;

receiving a request to access one of the peripheral devices;

determining whether the requested peripheral device is accessible;

if the request is a request to connect a computer to the requested peripheral device,

instantiating the connection class to create an object specific to the requested peripheral device,

using the instantiated object to cause a native driver of the requested peripheral device to execute, and

connecting, through the driver, the computer to the requested peripheral device; and

if the request is a request from the requested peripheral device to send data to the computer,

notifying the computer that the requested peripheral device has the data,

instantiating the connection class to create an object specific to the requested peripheral device,

using the instantiated object to cause the native driver of the requested peripheral device to execute,

connecting, through the driver, the computer to the requested peripheral device, and

sending the data from the requested peripheral device to the computer.

13. The method of claim 12, further comprising:

if the request is a request to disconnect the computer from the requested peripheral device,

using the instantiated object of the connection class to cause the native driver of the requested peripheral device to execute,

disconnecting, through the driver, the computer from the requested peripheral device, and

uninstantiating the connection class to delete the instantiated object.

14. A system comprising:

at least one peripheral device having associated therewith a native driver; and

a mobile computer configured to provide an interface used by an application to access the at least one peripheral device, the interface being independent of device-specific features of the at least one peripheral device.

15. The system of claim 14, wherein the computer is further configured, upon receiving a request, to use the interface to call a plurality of routines as a function of the device-specific features of the peripheral device and to cause the native driver, installed on the computer, to execute and control the peripheral device.

16. The system of claim 14, wherein the computer is further configured to receive a request from the application to access the peripheral device.

17. The system of claim 14, wherein the computer is further configured to receive a request from the peripheral device to provide data to the application.

18. The system of claim 14, wherein the computer is further configured,

upon receiving a request to access the at least one peripheral device, to use the interface to call a plurality of routines as a function of the device-specific features of the at least one peripheral device, and

upon receiving a request to access a second peripheral device, to use the interface to call the plurality of routines as a function of the device-specific features of the second peripheral device.

19. The system of claim 14, further comprising:

a second mobile computer, having the application ported thereto, configured to access a different peripheral device with the application, wherein the application on the second mobile computer uses the interface to access the different peripheral device without modifying the application.

20. The system of claim 14, wherein the mobile computer uses the interface to limit communication with the at least one peripheral device to one request at a time.